General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Carlos Alberto Armada
Kennedy Space Center
August 5th, 2008

Reviewed by:
James Stanley
Electromagnetic Compatibility Analysis Group VA-H3


(Signature)

Abstract

Armada, Carlos A.
Title of Project:
NASA/INSPIRE
NASA Center: Kennedy Space Center
Intern's Mentor: James Stanley
Internship Dates: June 16th – August 8th

During the eight weeks working at NASA, I was fortunate enough to work with the Expendable Launch Vehicle's (ELV) Electromagnetic Compatibility (EMC) Team, who is responsible for the evaluation and analysis of any EMI risk an ELV mission might face. This group of people concern themselves with practically any form of electromagnetic interference that may risk the safety of a rocket, a mission, or even people. Taking this into consideration, the group investigates natural forms of interference, such as lightning, to manmade interferences, such as antennas.

James Stanley, one of the members of the EMC Team, was assigned as my mentor for the eight weeks. To briefly explain what he does for NASA, Mr. Stanley creates simulations that can test the vulnerability and susceptibility of a device or material. For instance, at the moment Mr. Stanley is working on the Taurus II Certification and he has modeled a portion of the rocket in order to test its RF vulnerability, the obvious benefits from his simulations are cost-effectiveness and safety. Fortunately, Mr. Stanley was able to expose me to a bit of everything the EMC group does, research, software development, and even hands-on work.

The first project that I worked on was the investigation and then creation a small database for the electrical properties of about five materials. My mentor, James Stanley, creates electromagnetic simulations, and for each material that he tests he must have its properties. However, since the electrical properties of materials such as, epoxies and carbon fibers are so difficult to find, it simply is not practical and efficient for my mentor to look for the properties himself. For that reason, I was to look up properties such as permittivity, permeability, and electrical conductivity, so that James Stanley could effectively create his simulations. In addition to the electrical properties research I did, I also collected the thermal data for the same materials. Soon after I had started the research for Mr. Stanley, I realized how common the thermal properties were to come by. As a result, I approached Gary O'Neil, the thermal group lead, and proposed the idea of adding these properties to his existing database. Thrilled with the idea, I also began to collect data for Mr. O'Neil. Even though I spent many weeks talking to different companies and searching within NASA databases, it is difficult to summarize this type of research, however, some of the final products from the research can be seen in Appendix A.

Of the eight weeks, the development of software consumed the most amount of time. I was given an objective, and was told to solve it through the use of Matlab. Short for Matrix Laboratory, Matlab is a technical computing environment for high-performance computation and visualization. Best known for its ability to manipulate and use matrices, Matlab is able to create a multitude of visual representations ranging from bar graphs to 3D models. This summer, through the use of Matlab, three scripts where developed to meet specific objectives; two of which analyzed and manipulated data, and the other a 3D model.

Like mentioned above, the first two programs were developed to manipulate and analyze data; however, each very distinct in its purpose. The first program was assigned by James Stanley, and its objective was to analyze and plot the data produced by his Electromagnetic

Simulations. Although, before understanding how the program functions in depth, one must also have a general understanding on how Electromagnetic simulations work, and most importantly, how they output their data.

Although not a popular field of study, Electromagnetic modeling is an efficient and cost-effective way of testing the RF vulnerability/susceptibility of hardware and materials. Essentially, a simulation takes the work that must be done in the lab and converts it to the computer, which in-turn factors out the risk that could be encountered in the lab. Nonetheless, the program concerns itself mostly with the output file that the simulation creates, and not as much the process of the simulation. The data produced from running a simulation is documented into an out file, which has a format similar to a Notepad/WordPad document (Appendix B Figure 1). Difficult to explain without a computer, the file produced from running a simulation includes all the information, from the radiation pattern created from the antenna, to the values of the electric field strength. Within all this information of about two thousand lines, the program created only needs to manipulate and analyze one hundred lines of electric field values. Therefore, since the program must search for its data, rather than analyzing the entire document, it makes the commands more difficult to code and execute. After identifying the data that must be uploaded, the programs next objective is to define each column from a list of variables: x, y, z, Ex, Ey, Ez. Once identified, the program is to solve for the Electric field by using a combination of the Ex, Ey, and Ez variables and an equation resembling the Euclidean Norm: $\sqrt{Ex^2 + Ey^2 + Ez^2} = \mathrm{E}$. This equation defines the Electric field values, which in the program is assigned and plotted on the y-axis. The x-axis, although requiring less manipulation of the data, is far more complicated. Being one of the programs major features, the x-axis can vary in its values, and meaning. To explain this one must go back to how the data is outputted, for each x and y there is a corresponding E field value, however without going three dimensional there is no way to graph all three variables, therefore the program graphs the electric field values with relation to a constant and set of values. This still may be difficult to understand, but basically there are one hundred lines and for the x values the same ten values are repeated ten times, making one hundred lines. The y values are different in that the every ten lines the value changes, and there are ten values, also making one hundred lines. Now if that explanation made the manipulation process any clearer, one can see how each graph will have ten data points. For example, imagine that the user selects the x-values to be graphed in relation to the E-values with a y-constant of "-0.85", the program will only plot those x-values which correspond with the y-value of "-0.85", and like mentioned before each y-value is repeated ten times, which allows for twenty graphs per simulation (Appendix B Figure 2).

After programming this process, the next step was to create a user-interface window that would allow anyone to run the program, even without prior experience in Matlab. This is made possible through the use of Matlab's GUI add-on, which is very similar to a more familiar programming language like Visual Basic. Essentially the window runs the program as it would in the Matlab command window, but has more user-friendly buttons such as scroll menus and pop-up menus instead of typing the inputs into the command window. Another feature to the user-interface is that the graphs are plotted immediately onto the window, requiring no other program or pop-up window to run the program effectively and efficiently (Appendix B Figure 3).

Similar to the Electric Field script described above, the second program also manipulates and analyzes data but for a different purpose. This program, or as I call it "LDS Script", scans the data recorded from the lightning detection system and manipulates the data to solve for its minimum, maximum and peak-to-peak; once it completes those actions, it is up to the user on

what he would like to do, whether it be plotting the data or recording it into a file. This program has proven to be useful since the previous process of reading data from the EMC's lightning detection system was time consuming and horribly inefficient. As stated early in the paper, in the description of the EMC group, their field of analysis is highly concerned with all types of natural and manmade forms of interference and since Kennedy Space Center is located in the lightning capital of the Unites States, it is inevitable that lightning be looked at further in depth. Which leads to the lightning detection system located on Complex 17 Pad B, being just one of the many lightning detection systems on Kennedy Space Center, this specific unit is under the control of the EMC group and therefore, it is the responsibility of the group to monitor for lightning strikes in the area and record it for research purposes.

How this Lightning Detection System, LDS for short, works, is that it has two sensors, an electric field sensor, and a Pearson coil which work together to record the induced current from a lightning strike. Now when there is a lightning strike within the area of Complex 17 the lightning's electric field triggers the electric field sensor, which in-turn tells the LDS to begin recording. During this short interval, the LDS will take in all the data in microseconds that the Pearson coil is reading, which comes from the induced current running through T-0 umbilical. The Pearson coil is capable of reading a current because the lightning strike creates a magnetic field, and as learned in Physics class, can create an induced current on a closed circuit. It is through this data that the group can measure the intensity of the strike (Appendix C Figure 1).

Comparable to the .out file produced by the electromagnetic simulations, the LDS when recording will insert all the data into an .in file. Fortunately, unlike the first program, the information in the file is solely the numerical data, requiring no command to locate the data. Therefore by simply selecting the file to analyze, the program will immediately begin to manipulate and crunch the numbers. This script begins by identifying each column as a variable in the following order, time (in seconds), current, vertical electric field, horizontal electric field, and external electric field. The program then averages the first hundred values in order to create a value for the noise in the recording. Next, the program factors out the noise level in the recording and then average every ten data points together, converting a one by two thousand matrix to a one by two hundred matrix. This process clears up the graphs, that otherwise would be too messy for any user to read/analyze. After writing the code to manipulate the data, as explained above, the final step, like before, was to create a user interface window through GUI so that anyone could effectively use the program. As a result, I created a simple window that has a scroll menu with all the files that a user can analyze and two buttons, the first to trigger the program to analyze the data and write it into an .out file, and the second button to open another window displaying the selected files current v. time graph. Specifically, the first button writes the analyzed file's minimum, maximum, and peak-to-peak for later analysis, and the second button would do the same, but instead of saving the data to a file, it was for a quick analysis in real-time (Appendix C Figure 2).

The third and final script written with Matlab has no similarities what so ever to the first two programs, rather than being logic based, this script revolves around mathematics, due to the nature of the project. The objective of the program was to as accurately as possible create a 3D model of the Catenary System found on Space Launch Complex 41. Unfortunately, resources were limited and the only aids used were Google Earth and one image of the catenary system (Appendix D Figure 1, 2). So, to begin the project, I used Google Earth to calculate the distance between each tower. I then imported the image of the catenary system into Matlab and solved for the amount of pixels per foot, consequently, this allowed for the solving of the vertex. Once the

vertex and two points were known, the first cable's parabolic function was solved for, and the rest of the model was solved through the use of proportions. Possibly not being the best way to solve for each cable's function, proportions were the only feasible way of creating the 3D model, especially due to the lack of recourses. Regardless of the limitations, the model's functions were then graphed onto one plot, and eventually resembled the system found on SLC-41 with about 2 feet of marginal error. Hard to put into words, the images justify the model more than the description does (Appendix D Figure 3).

The final project that I was fortunate enough to work on was the $1/5^{th}$ Scaled Fairing Experiment. Inherited from a previous thermal experiment, this project involved taking a $1/5^{th}$ Scaled Fairing experiment and testing its RF vulnerability/susceptibility. Similar to how a simulation works on the computer, this experiment uses two dual-horn antennas, which are located on the inside of the fairing, and they emit a specified frequency. By comparing the values inputted to the data outputted, the EMC team can gauge how the fairing responds to different amplitudes of RF energy. The results from this experiment depend on what is located inside of the fairing, and this is where I was able to contribute to the experiment. Even though the experiment had been started previous to my arrival, I was able to assist by adding a thin layer of Kapton, a thermal blanket, to the interior of the fairing. This is useful because the point of this experiment is to replicate, as close as possible, a real Fairing. Within an actual fairing there are a multitude of materials that layer the inside cavity, and therefore, in our experiment, one by one, we have been adding a new layer of material. At this point our replica has a layer of aluminum foil and Kapton, but eventually it will have four layers: aluminum foil, Kapton, melamine foam, and another layer of Kapton. Through this experiment, the EMC group hopes to expand their current understanding of the fairing, as well as compare their computer simulations to an actual test.

Throughout my experience here at NASA, I was lucky enough to have a wonderful mentor and work with great and knowledgeable people. After working here for eight weeks, I have learned valuable researching techniques, I have learned how to develop programs, and most importantly, I have been learned about and been exposed to a various amount of technical fields. I would like to acknowledge: Fred Becker, Ron Brewer, Laura Brumm, TD Doan, Rick Iacabucci, Andrew Lash, David Piryk, Linda Read-Ross, Steve Chance, Kevin Clinton, John Giles, Priscilla Moore, Gary O'Neil, Dawn Trout, Lisa Valencia, Roger Spears, Richard Adams, Jim Gerard, and most importantly, my mentor, James Stanley.

Appendix A

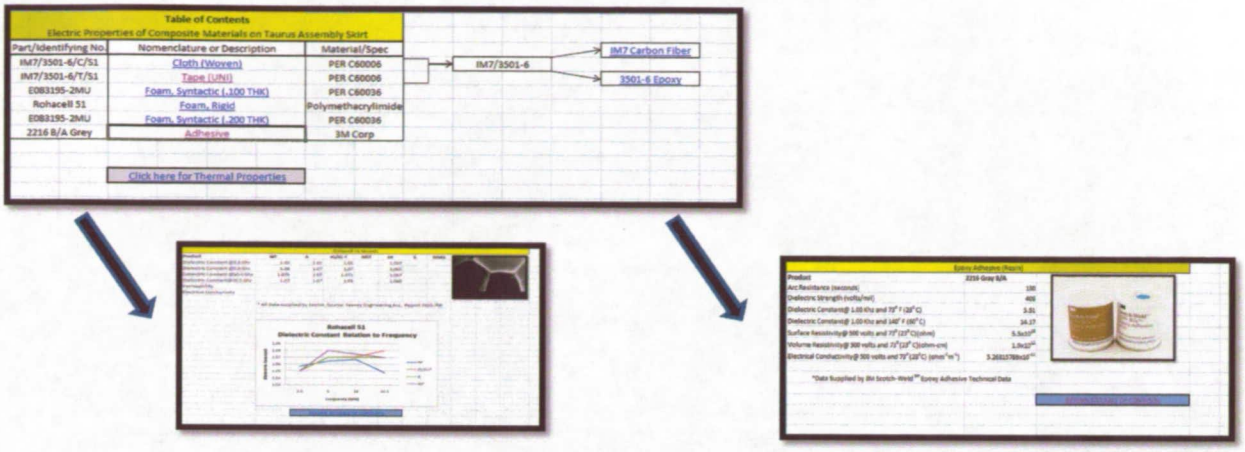


Figure 1. Portion of Electrical Properties Database.

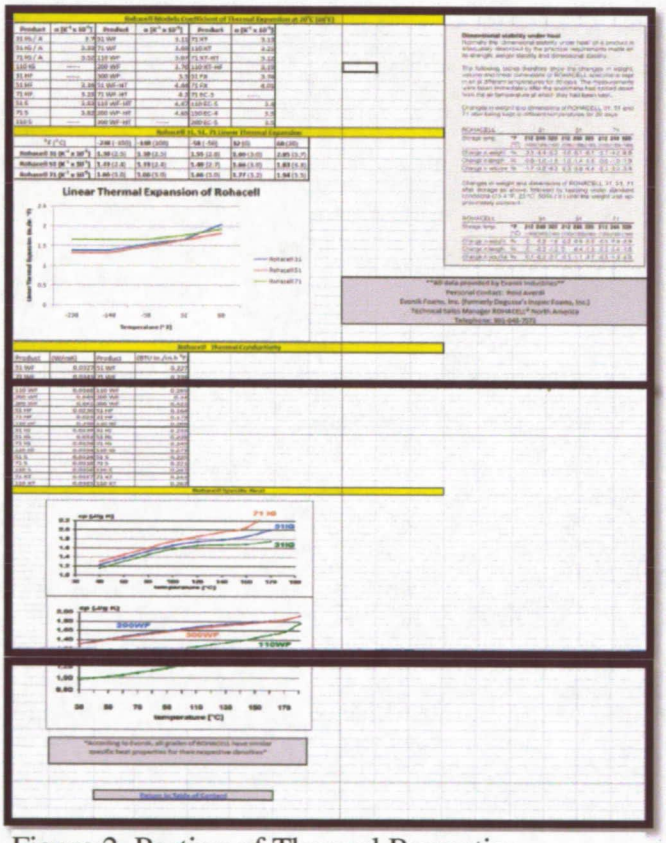
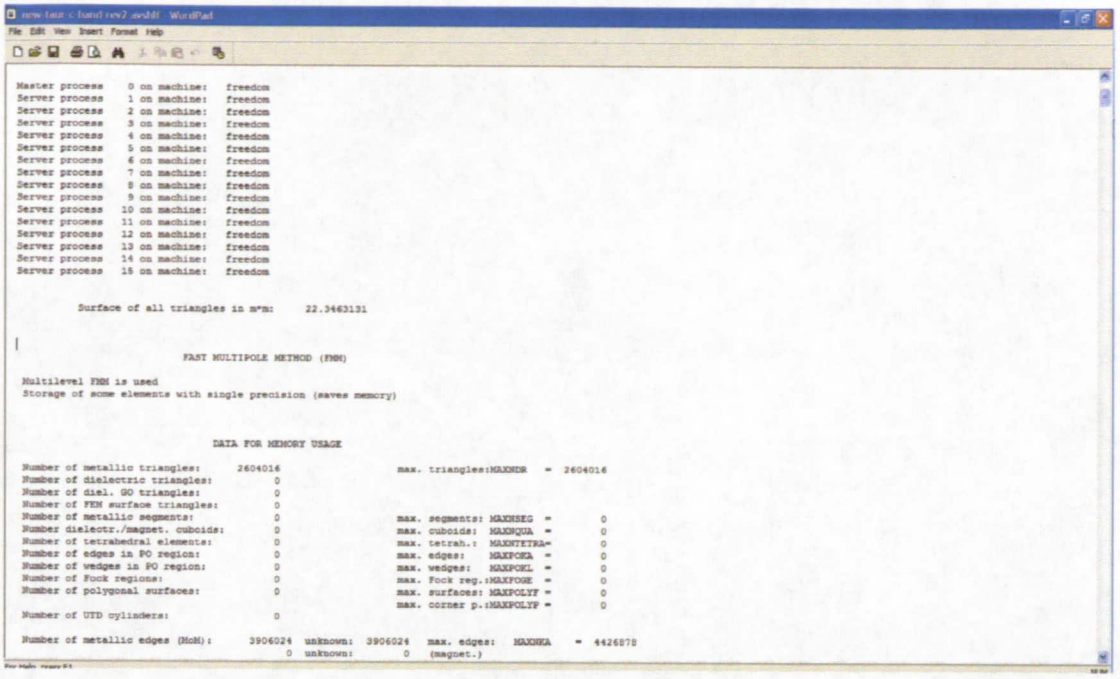


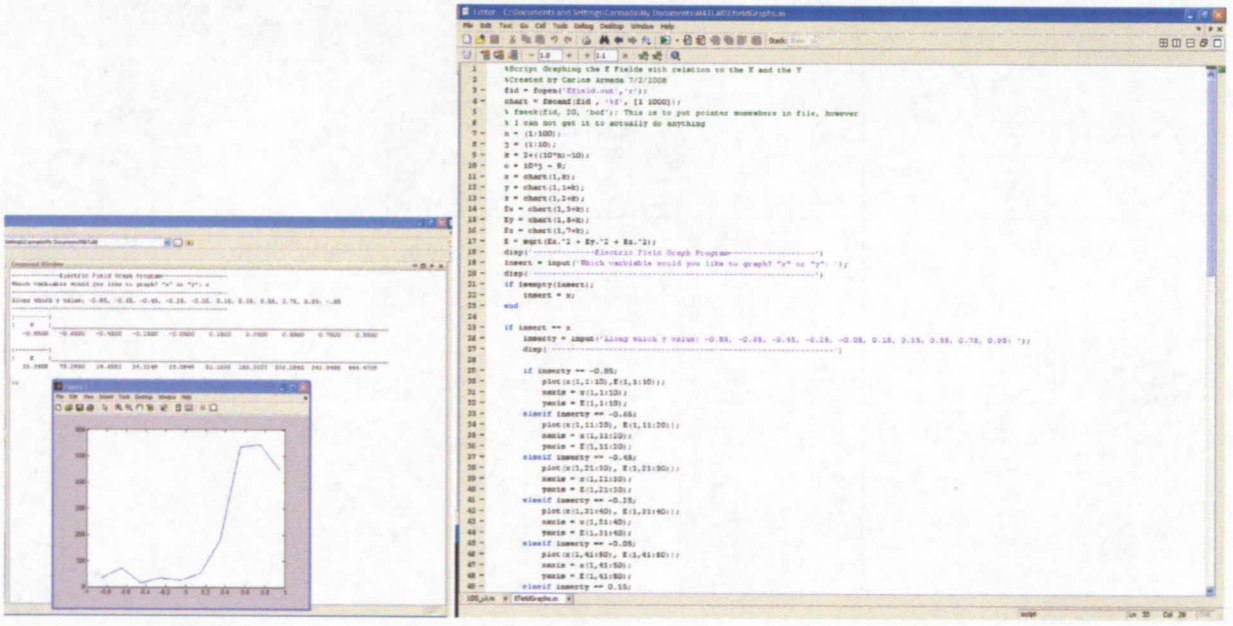
Figure 2. Portion of Thermal Properties Database.

Appendix B



```
Master process     0 on machine:   freedom
Server process     1 on machine:   freedom
Server process     2 on machine:   freedom
Server process     3 on machine:   freedom
Server process     4 on machine:   freedom
Server process     5 on machine:   freedom
Server process     6 on machine:   freedom
Server process     7 on machine:   freedom
Server process     8 on machine:   freedom
Server process     9 on machine:   freedom
Server process    10 on machine:   freedom
Server process    11 on machine:   freedom
Server process    12 on machine:   freedom
Server process    13 on machine:   freedom
Server process    14 on machine:   freedom
Server process    15 on machine:   freedom

        Surface of all triangles in m**m:      22.3463131

                        FAST MULTIPOLE METHOD (FMM)

 Multilevel FMM is used
 Storage of some elements with single precision (saves memory)

                        DATA FOR MEMORY USAGE

 Number of metallic triangles:       2604016                 max. triangles:MAXNDR   =  2604016
 Number of dielectric triangles:           0
 Number of diel. GO triangles:             0
 Number of FEM surface triangles:          0
 Number of metallic segments:              0                 max. segments: MAXNSEG  =        0
 Number dielectr./magnet. cuboids:         0                 max. cuboids:  MAXNQUA  =        0
 Number of tetrahedral elements:           0                 max. tetrah.:  MAXNTETRA=        0
 Number of edges in PO region:             0                 max. edges:    MAXPOEA  =        0
 Number of wedges in PO region:            0                 max. wedges:   MAXPOKL  =        0
 Number of Fock regions:                   0                 max. Fock reg.:MAXFOGE  =        0
 Number of polygonal surfaces:             0                 max. surfaces: MAXPOLYF =        0
                                                             max. corner p.:MAXPOLYP =        0
 Number of UTD cylinders:                  0

 Number of metallic edges (MoM):     3906024  unknown:  3906024   max. edges:   MAXNEA   =  4426878
                                           0  unknown:        0   (magnet.)
```

Figure 1. Output from Electromagnetic Simulation in .out file, very similar to Notepad/WordPad document.



```
%Script Graphing the E Fields with relation to the X and the Y
%Created by Carlos Armenta 7/2/2008
fid = fopen('Efield.out','r');
chart = fscanf(fid , '%f', [1 1000]);
% fseek(fid, 20, 'bof'); This is to put pointer somewhere in file, however
% I can not get it to actually do anything
n = (1:100);
j = (1:10);
k = 1+(10*n)-10;
o = 10*j - 9;
x = chart(1,k);
y = chart(1,1+k);
z = chart(1,2+k);
Ex = chart(1,3+k);
Ey = chart(1,4+k);
Ez = chart(1,7+k);
E = sqrt(Ex.^2 + Ey.^2 + Ez.^2);
disp('------------Electric Field Graph Program------------')
insert = input('Which variable would you like to graph? "x" or "y": ');
disp('---------------------------------------------------')
if isempty(insert);
    insert = x;
end

if insert == x
    inserty = input('Along which Y value: -0.05, -0.04, -0.03, -0.02, -0.01, 0.1, 0.15, 0.2, 0.25, 0.3: ');
    disp('---------------------------------------------------')

    if inserty == -0.05;
        plot(x(1,1:10),E(1,1:10));
        xaxis = x(1,1:10);
        yaxis = E(1,1:10);
    elseif inserty == -0.04;
        plot(x(1,11:20), E(1,11:20));
        xaxis = x(1,11:20);
        yaxis = E(1,11:20);
    elseif inserty == -0.03;
        plot(x(1,21:30), E(1,21:30));
        xaxis = x(1,21:30);
        yaxis = E(1,21:30);
    elseif inserty == -0.02;
        plot(x(1,31:40), E(1,31:40));
        xaxis = x(1,31:40);
        yaxis = E(1,31:40);
    elseif inserty == -0.01;
        plot(x(1,41:50), E(1,41:50));
        xaxis = x(1,41:50);
        yaxis = E(1,41:50);
    elseif inserty == 0.1;
```

Figure 2. The Matlab program before implementation of GUI. Output of the program shown on the left in Matlab Command Window. The purple lines denote the inputs a user can make to manipulate the x-axis.

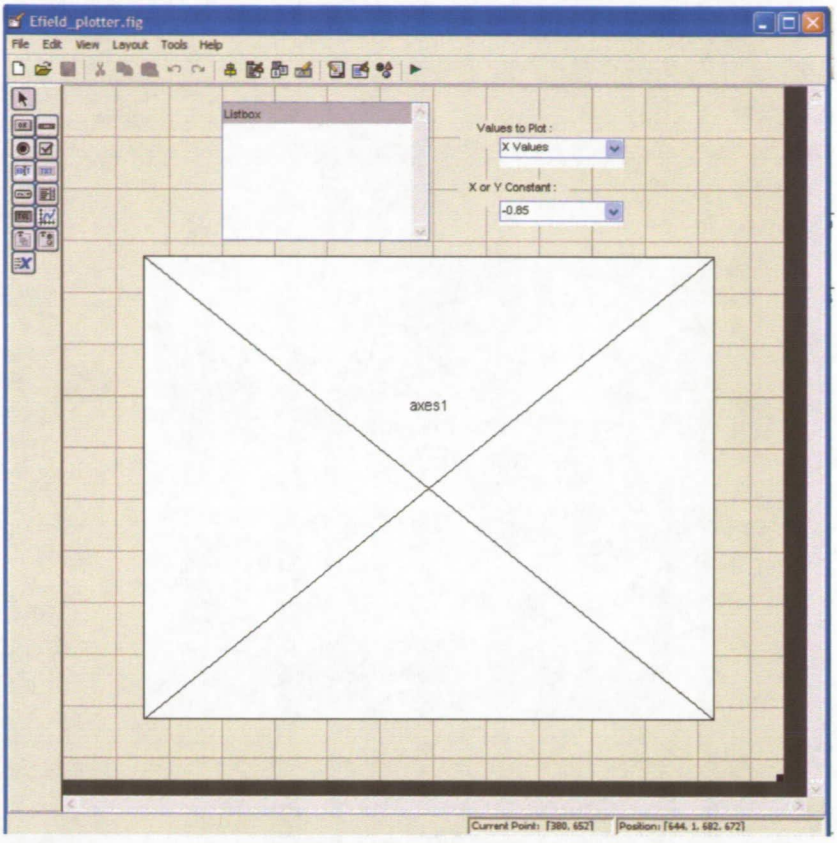


Figure 3. GUI Window Interface Construction. In addition to the code, this window constructer allows the final product to function. As displayed, there are scroll menus, pop-up menus, and even a graph.

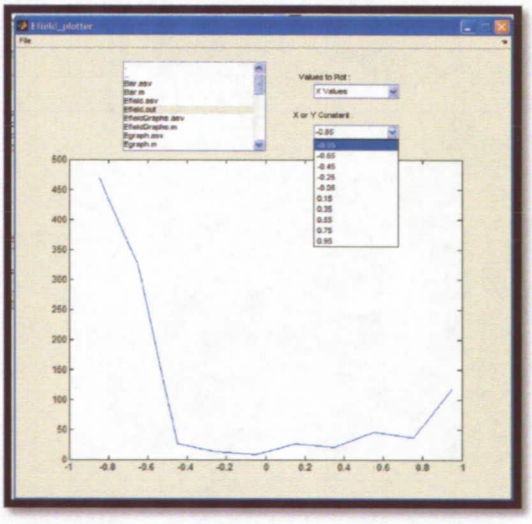


Figure 4. End result of Electromagnetic Modeling Analyzer Program.

Appendix C

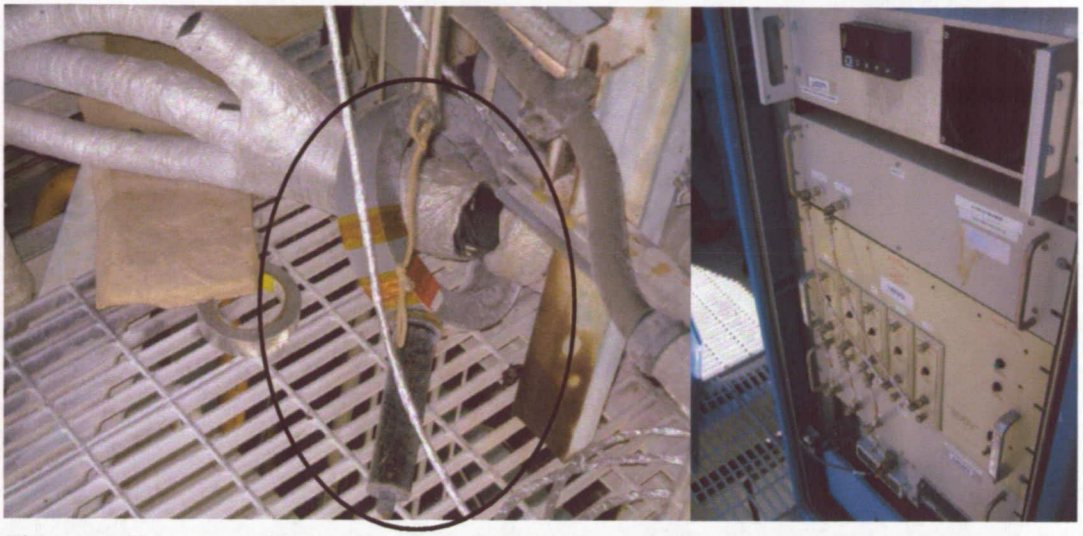


Figure 1. Pearson coil (Left) and LDS Rack (Right) located on 8th floor of Complex 17 Pad B.

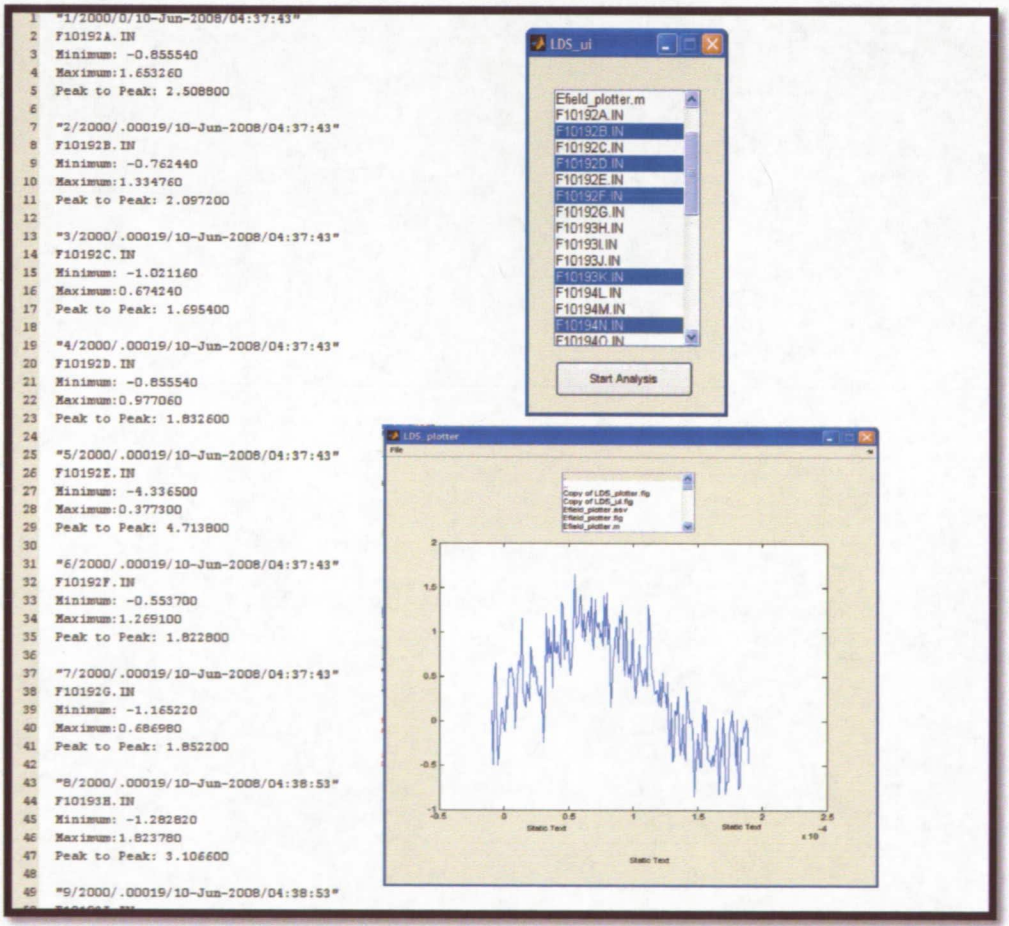


Figure 2. End Result of the program. Both GUI Windows and the end product of running analysis (Text on the left of the min/max/peak-to-peak).